DBMS Project Report

PES University

Database Management Systems

UE18CS252

Submitted By

PES2201800415 AKASH YADAV

A Bus Reservation database System has been implemented here. The functions of this database system are to maintain records of the various transactions that happen, from the booking of ticket to the departure of the passenger. The various entities used are Bus station, Bus, Bus company, Seat, Passenger details, the fare of the trip and the User details.

The design of the database was done using an ER model. A schema and an EER diagram have been depicted for a better understanding. Various relations and tables have been shown and how each of them are related. The cardinality of these relationships is also given. The tables were made using the DDL commands.

Triggers were used to counter certain constraints like the base fare being updated incorrectly or checking if the tax to be collected is entered correctly. Audit – triggers were also displayed.
Queries were used to display the details of the passengers and the fare prices. They were also used to display common values in two tables that matched a certain condition.

This database is capable of automating all the basic processes of a reservation system. The **Intra Indian Bus Transit System (IIBTS)** created is capable of managing large transactions thus improving the current scenario where manual transactions are done. It has its limitations but is a good approach to automating the reservation system.

Introduction

The mini-world being depicted here is of a Bus Reservation System.

Bus reservation system is one of the most used database system in the world. It is an example of Transaction processing systems. **Transaction processing systems** are systems with large databases and hundreds of concurrent users executing database transactions. These systems require high availability and fast response time for hundreds of concurrent users.

The database being built here is an **Intra Indian Bus Transit System (IIBTS)** wherein all the data, right from the user details to the bus details are stored. In this project, the database part of the whole system is dealt with, which include insertions, deletions and updates to the database.

The basic transactions of this database is that there is a *User* who books the *ticket* , *the bus company* receives the booking, collects the *fare* and then allocates a *seat* in one of its buses , taking the source and destination in mind. The *Bus stations* then give access to these *buses* that have *passengers* in them. *A traveller itinerary* is then issued for every passenger, which has the important details of their *Bus Trip*.

Some of the important entities are as follows:

BUS_STATION – A transit point for all buses. It is uniquely identified by the Bus Station Code. The other important attributes are Bus Station name, Location, city, state, country and zip.

BUS_COMPANY – The owner of multiple buses having a unique Company ID and a company name.

BUS - The most important entity having a unique Bus No. The entity also has attributes like its seating capacity and its company name.
Passenger – This entity holds details about the bus commuters. It is uniquely identified by an ID.

Bus trip – This entity holds details about the PNR and other details of the bus trip.

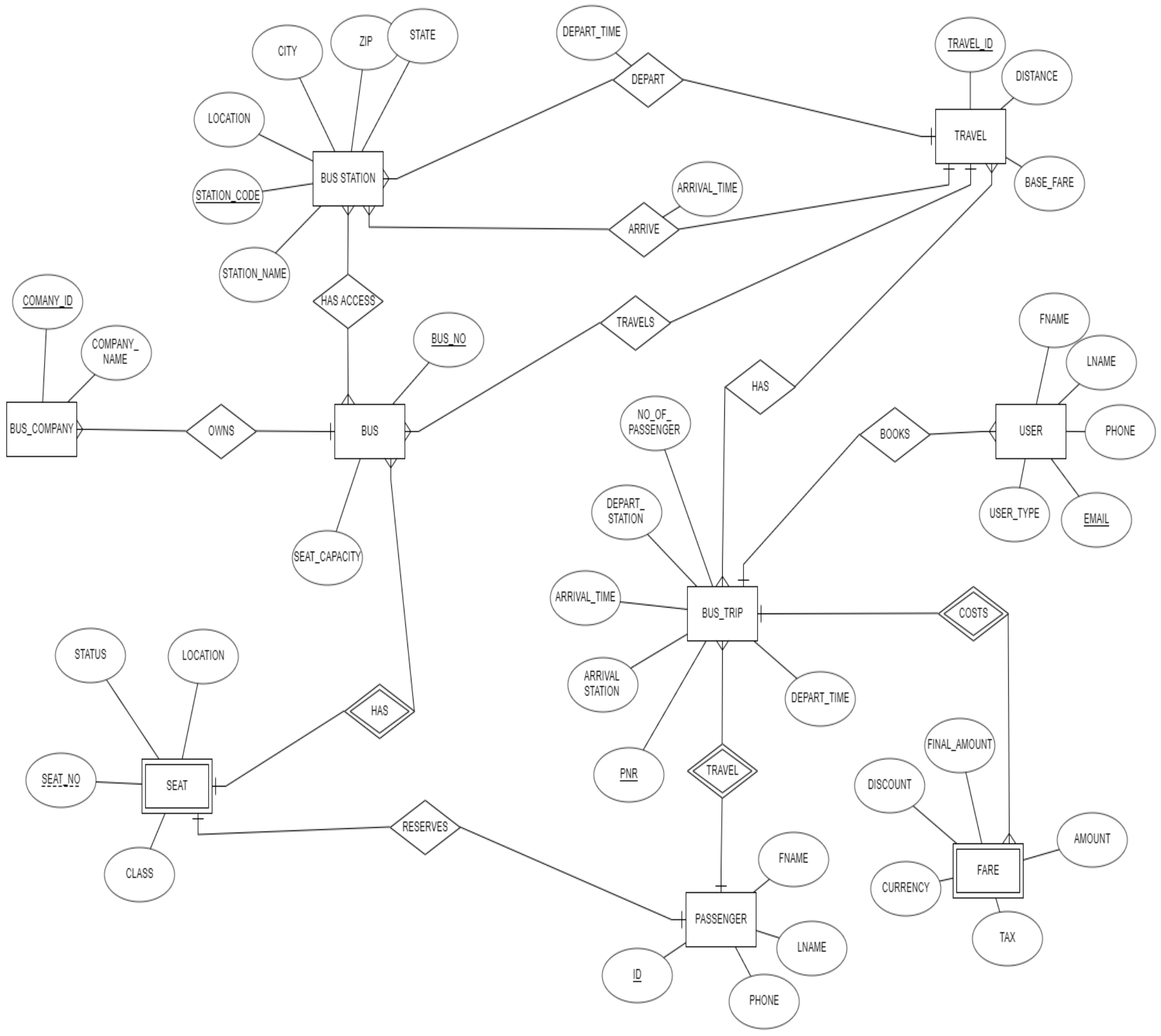
Certain Assumptions have been made to make the database efficient, which are listed below.

Assumptions

1) The access to a Bus Station is dependent on the Bus and not the Bus Company. A Bus Station can give access to multiple buses and a bus can have access to multiple Bus Stations.
2) Every Bus must belong to a Bus company. So a full participation of Bus is justified. A company can have multiple buses but a bus must belong to one company.
3) Each seat belongs to a bus and bus has multiple seats. Without a bus, seat cannot exist. Seat is a dependent attribute (bus).
4) Each passenger must have a seat. However, seats can be empty so it’s not necessary that every seat belongs to a passenger but the opposite is possible. So a full participation of passenger is justified.
5) A user can book tickets for multiple passengers and is an independent entity unlike passenger.

Data Model

Taking the assumptions in mind, the following Entity Diagram relation was incorporated.



One-to-One Binary Relationships (1:1)

- SEAT-reserves-PASSENGER
- BUS_TRIP - costs – FARE

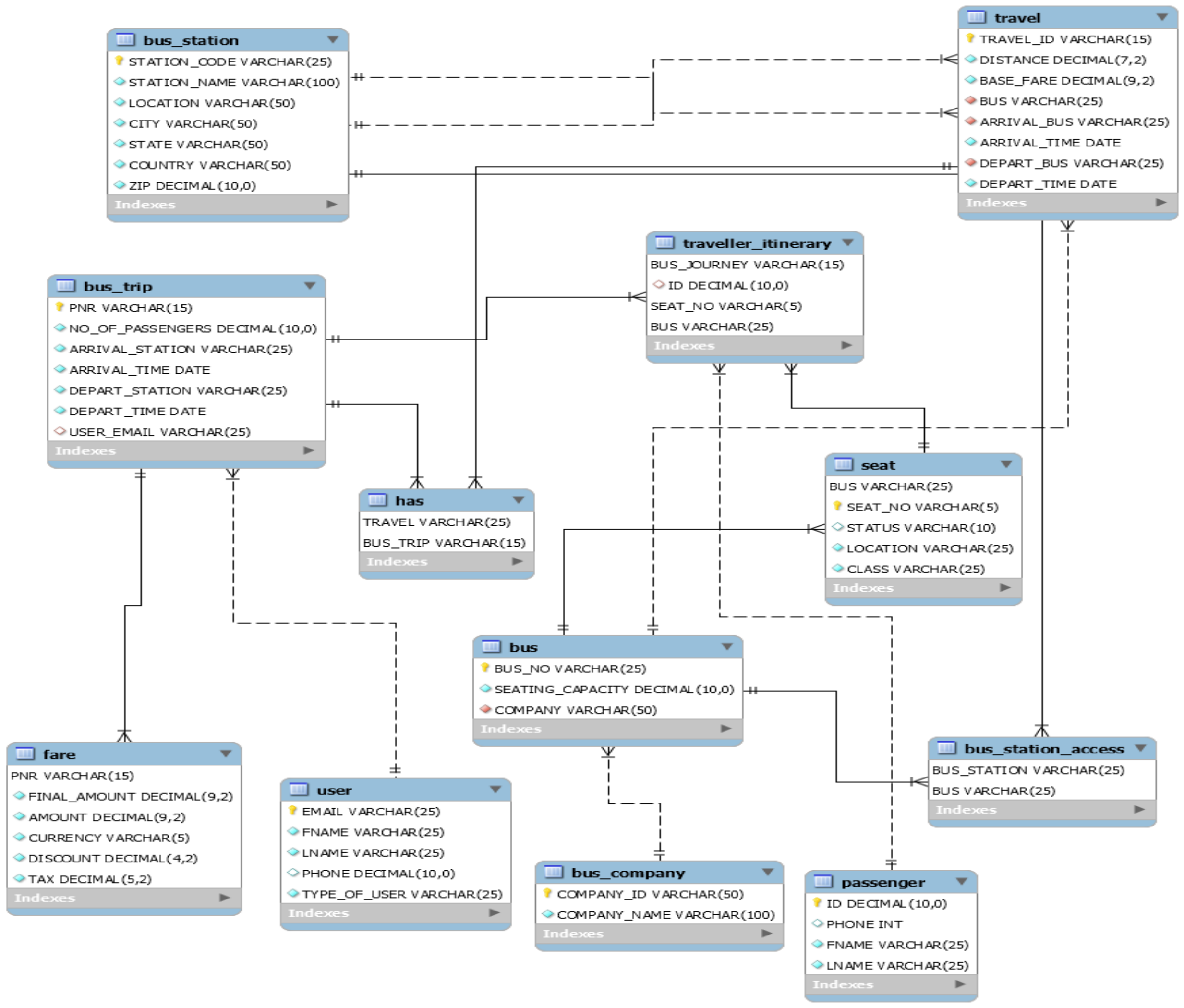
One-to-Many Binary Relationships(1:N)

- BUS - travels - TRAVEL
- BUS_STATION - arrive - TRAVEL
- BUS_STATION - depart – TRAVEL
- BUS_COMPANY - owns - BUS
- BUS - has – SEATS
- BUS_TRIP - travel – TRAVELLER
- USER - books - BUS_TRIP

Many-to-Many Binary Relationships(N:M)

- BUS_STATION - access - BUS
- BUS_TRIP - has - TRAVEL

The Schema that we get is:



The above schema diagram shows the relations between the entities and also the type of variables that are used for the attributes. Constraints such as primary key, referential integrity and NOT NULL constraints have been added to the formation of the table**. Check constraint** has also been added to check whether the given phone number is a 10 digit number or not.

Varchar is used the most as it can accommodate both letters and numbers. DATE variable is also used for the Date of journey and other attributes.

FD and Normalization

The Functional Attributes of the tables are given below:

BUS_STATION Table

Bus_station_code -> {bus_station_name,location,city,state,zip,country}

BUS_COMPANY Table

Company_ID -> {company_name}

SEAT Table

(Bus,Seat_no) -> {Status,Class}

TRAVELLER_ITINEARAY Table

(ID,Seat_no)->{Bus}
Bus_Journey->{Bus}

PASSENGER Table

ID -> {Fname,Lname,Phone}

BUS_TRIP Table

PNR->{No_of_passengers , arrival_station , arrival_time , depart_station , depart_time , user_email}

FARE Table

PNR ->{ Amount,tax,discount,Final_amount,currency}

USER Table

EMAIL ->{Fname,Lname,Phone,Type_of_user}

TRAVEL Table

Travel_Id -> {distance, base fare , bus, arrival_station , arrival_time , depart_station , depart_time}

As the schema was constructed by converting the ER Model , the tables that we have gotten are already normalized.

But to understand the violation of 1st normal form , we'll take an example, In the USER table if the PHONE attribute was composite or a multi valued attribute then the atomicity of that table would not exist. This would have been the violation of the 1st form wherein a cell in the column of a table takes more than one value.

To understand the 2nd normal form , let's take the example of the table BUS. Let's add a new attribute called Bus_registration number which is unique to every bus and it is different from the Bus_no.(First of all the 1st NF should be satisfied) .When we add this column the table now has a candidate key compromising of the Bus_no and the Bus_registration number. This leads to partial dependency of other attributes on these two primary keys. This leads to the violation of the 2nd form , to resolve this we have to break the table into such that the 2nd NF is not violated.

The 3rd NF deals with transitive dependency (A->B and B->C are two FDs then A->C is called transitive dependency).To understand this lets take an example of the Bus_Company and the Bus table. When we combine these tables we see that Company_name->Company_ID and CompanyId->Bus_no which proves that Company_name->Bus_no.('->' means depends), this violates the 3rd NF because transitive dependency of Company_name on Bus_no exists. To resolve this , the tables are divided.

Testing for lossless join → When a natural join is performed on both the tables , it yields the original table without any loss of data .It has a lossless join

These are the final tables that have been created for the normalized relational schema mentioned in the above schema. The primary key of each table is chosen carefully to ensure that any other attribute of the table is not dependent on entity other than the primary key. The table is as follows:

- BUS_STATION – Bus_station_Code
- BUS_COMPANY - Company_ID
- BUS - BUS_NO
- BUS_STATION_ACCESS - Bus, Bus_station
- SEAT - Bus, Seat_no
- PASSENGER - ID
- TRAVELLER_ITINERARY – BUS_JOURNEY, ID
- BUS_TRIP - PNR
- FARE - PNR
- USER - Email
- TRAVEL - Travel_id

DDL

Tables are created below , check constraints are used wherever applicable

BUS_STATION

```
CREATE TABLE BUS_STATION(
STATION_CODE VARCHAR(25),
STATION_NAME VARCHAR(100) NOT NULL,
LOCATION VARCHAR(50) NOT NULL,
CITY VARCHAR(50) NOT NULL,
STATE VARCHAR(50) NOT NULL,
COUNTRY VARCHAR(50) NOT NULL,
ZIP NUMERIC NOT NULL,
CONSTRAINT BUS_STATION_PK PRIMARY KEY (STATION_CODE)
);
```

BUS_COMPANY

```
CREATE TABLE BUS_COMPANY(
COMPANY_ID VARCHAR(50) ,
COMPANY_NAME VARCHAR(100) NOT NULL,
CONSTRAINT COMPANY_PK PRIMARY KEY (COMPANY_ID)
);
```

BUS

```
CREATE TABLE BUS(
BUS_NO VARCHAR(25) ,
SEATING_CAPACITY NUMERIC NOT NULL,
COMPANY VARCHAR(50) NOT NULL,
CONSTRAINT BUS_PK PRIMARY KEY (BUS_NO),
CONSTRAINT BUS_COMPANY_FK FOREIGN KEY (COMPANY)
REFERENCES BUS_COMPANY(COMPANY_ID)
);
```

BUS_STATION_ACCESS

```
CREATE TABLE BUS_STATION_ACCESS(
BUS_STATION VARCHAR(25) NOT NULL,
BUS VARCHAR(25) NOT NULL,
CONSTRAINT ACCESS_PK PRIMARY KEY (BUS_STATION,BUS),
CONSTRAINT ACCESS_PORT_FK FOREIGN KEY (BUS_STATION)
REFERENCES BUS_STATION(STATION_CODE)
ON DELETE CASCADE,
CONSTRAINT ACCESS_BUS_FK FOREIGN KEY (BUS)
REFERENCES BUS(BUS_NO)
ON DELETE CASCADE
);
```

SEAT

```
CREATE TABLE SEAT(
BUS VARCHAR(25) NOT NULL,
SEAT_NO VARCHAR(5) NOT NULL,
STATUS VARCHAR(10) DEFAULT 'TRUE',
LOCATION VARCHAR(25) NOT NULL,
CLASS VARCHAR(25) NOT NULL,
Check (CLASS in(‘SLEEPER’,’GENERAL’,AC’)
CONSTRAINT SEAT_PK PRIMARY KEY (BUS,SEAT_NO),
CONSTRAINT SEAT_BUS_FK FOREIGN KEY (BUS)
REFERENCES BUS(BUS_NO)
ON DELETE CASCADE
);
```

USER

```
CREATE TABLE USER(
EMAIL VARCHAR(25) ,
FNAME VARCHAR(25) NOT NULL,
LNAME VARCHAR(25) NOT NULL,
PHONE NUMERIC ,
TYPE_OF_USER VARCHAR(25) NOT NULL,
CONSTRAINT USER_PK PRIMARY KEY (EMAIL)
);
ALTER TABLE USER
ADD constraint sd_chk CHECK (LENGTH(PHONE) = 10);
```

BUS_TRIP

```
CREATE TABLE BUS_TRIP(
PNR VARCHAR(15) ,
NO_OF_PASSENGERS NUMERIC NOT NULL,
ARRIVAL_STATION VARCHAR(25) NOT NULL,
ARRIVAL_TIME DATE NOT NULL,
DEPART_STATION VARCHAR(25) NOT NULL,
DEPART_TIME DATE NOT NULL,
USER_EMAIL VARCHAR(25) ,
CONSTRAINT BUS_TRIP_PK PRIMARY KEY
(PNR),
CONSTRAINT BUS_TRIP_USER_FK FOREIGN KEY
(USER_EMAIL) REFERENCES USER(EMAIL)
ON DELETE SET NULL
);
```

FARE

```
CREATE TABLE FARE(
PNR VARCHAR(15) ,
FINAL_AMOUNT DECIMAL(9,2) NOT NULL,
AMOUNT DECIMAL(9,2) NOT NULL,
CURRENCY VARCHAR(5) NOT NULL,
DISCOUNT DECIMAL(4,2) NOT NULL,
TAX DECIMAL(5,2) NOT NULL,
CONSTRAINT FARE_PK PRIMARY KEY (PNR),
CONSTRAINT FARE_FLT_TRIP_FK FOREIGN KEY
(PNR) REFERENCES BUS_TRIP(PNR)
ON DELETE CASCADE
);
```

PASSENGER

```
CREATE TABLE PASSENGER(
ID NUMERIC,
PHONE INTEGER,
FNAME VARCHAR(25) NOT NULL,
LNAME VARCHAR(25) NOT NULL,
CONSTRAINT PASSENGER_PK PRIMARY KEY (ID)
);
```

TRAVELLER_ITINERARY

```
CREATE TABLE TRAVELLER_ITINERARY(
BUS_JOURNEY VARCHAR(15) NOT NULL,
ID VARCHAR(15),
SEAT_NO VARCHAR(5),
BUS VARCHAR(25),
CONSTRAINT TRAVELLER_PKK PRIMARY KEY
(BUS_JOURNEY,SEAT_NO,BUS),
CONSTRAINT TRVLR_BUS_TRIP_FK FOREIGN KEY
(BUS_JOURNEY)
REFERENCES BUS_TRIP(PNR)
ON DELETE CASCADE,
CONSTRAINT TRVLR_SEAT_FK FOREIGN KEY(BUS,SEAT_NO)
REFERENCES SEAT(BUS,SEAT_NO)
);
```

TRAVEL

```
CREATE TABLE TRAVEL(
TRAVEL_ID VARCHAR(15) ,
DISTANCE DECIMAL(7,2) NOT NULL,
BASE_FARE DECIMAL(9,2) NOT NULL,
BUS VARCHAR(25) NOT NULL,
ARRIVAL_BUS VARCHAR(25) NOT NULL,
ARRIVAL_TIME DATE NOT NULL,
DEPART_BUS VARCHAR(25) NOT NULL,
DEPART_TIME DATE NOT NULL,
CONSTRAINT HOP_PK PRIMARY KEY (TRAVEL_ID),
CONSTRAINT HOP_BUS_FK FOREIGN KEY (BUS)
REFERENCES BUS(BUS_NO),
CONSTRAINT HOP_ARVL_STN_FK FOREIGN KEY(ARRIVAL_BUS)
REFERENCES BUS_STATION(STATION_CODE),
CONSTRAINT HOP_DRPT_BUS_FK FOREIGN KEY(DEPART_BUS)
REFERENCES BUS_STATION(STATION_CODE)
);
```

HAS

```
CREATE TABLE HAS(
TRAVEL VARCHAR(25) NOT NULL,
BUS_TRIP VARCHAR(15) NOT NULL,
CONSTRAINT HP_HAS_BUS_PK PRIMARY KEY
(TRAVEL,BUS_TRIP),
CONSTRAINT HOP_HAS_FK FOREIGN KEY (TRAVEL)
REFERENCES
TRAVEL(TRAVEL_ID),
CONSTRAINT BT_HAS_FK FOREIGN KEY(BUS_TRIP)
REFERENCES BUS_TRIP(PNR)
ON DELETE CASCADE
);
```

Cascades have been used wherever applicable, to remove irregularity in the data when it is updated.

Triggers

Create a trigger which the sets the Tax to 100 if the updated value in the table is LESSER than 100.

```
CREATE TRIGGER updcheck1 BEFORE UPDATE ON FARE
FOR EACH ROW
BEGIN
IF NEW.TAX>100 THEN
SET NEW.TAX = NEW.TAX;
ELSE
SET NEW.TAX = 100;
END IF;
END;
```

Create a trigger which the sets the base_fare to 500 if the updated value in the table is greater than 500.

```
CREATE TRIGGER updcheck2 BEFORE UPDATE ON travel
FOR EACH ROW
BEGIN
IF NEW. base_fare > 500 THEN
SET NEW. base_fare = NEW. base_fare;
ELSE
SET NEW. base_fare = 500;
END IF;
END;
```

SQL Queries

1. Display the first name and the last name of the passenger whose BUS_JOURNEY id =’h17dec’.

```
SELECT FNAME, LNAME
FROM PASSENGER
WHERE ID IN (SELECT ID FROM TRAVELLER_ITINERARY
WHERE BUS_JOURNEY='h17dec');
```

2. Display the email and the number of passengers whose user type is ‘Normal’. Group them by their User emails.

```
SELECT USER_EMAIL, COUNT(NO_OF_PASSENGERS)
FROM BUS_TRIP
WHERE USER_EMAIL IN (SELECT EMAIL
FROM user WHERE TYPE_OF_USER ='NORMAL')
GROUP BY USER_EMAIL;
```

3. Display all the common values from Bus_trip and fare table where their PNR values match.

```
SELECT * FROM BUS_TRIP INNER JOIN FARE ON
BUS_TRIP.PNR=FARE.PNR;
```

4. Display first name, last name and phone numbers of all those passengers whose BUS_JOURNEY id is h17dec or i17dec.

```
SELECT FNAME, LNAME,PHONE FROM PASSENGER WHERE ID IN(SELECT ID
FROM TRAVELLER_ITINERARY WHERE BUS_JOURNEY='h17dec')
UNION
SELECT FNAME, LNAME,PHONE FROM PASSENGER WHERE ID IN(SELECT ID FROM
TRAVELLER_ITINERARY WHERE BUS_JOURNEY='i17dec');
```

5. Display all the Bus numbers, the seating capacity of all the buses and their company names which are of the same company id.

```
SELECT BUS_NO,SEATING_CAPACITY,
Company_name
FROM BUS_company,bus
WHERE company_id = company;
```

6. Write a query to find the number of guests travelling along with the person who has booked the tickets and is not an agent and show the departure and the arrival date.

```
SELECT
USER_EMAIL,
   NO_OF_PASSENGERS,
   DEPART_TIME,
   ARRIVAL_TIME
FROM
BUS_TRIP
WHERE
USER_EMAIL IN
   (
SELECT
EMAIL
FROM
USER
WHERE
TYPE_OF_USER ='NORMAL'
)
GROUP BY
USER_EMAIL;
```

7. Write a query to find the total price of the person who is travelling to Bengaluru by arranging them from highest to lowest price.

```
SELECT
BUS_TRIP_ID,
   CURRENCY,
   FINAL_AMOUNT AS FINAL_AMOUNT
FROM
FARE
WHERE
PNR IN
   (
SELECT
PNR
FROM
BUS_TRIP
WHERE
ARRIVAL_STATION = 'BLR'
)
ORDER BY
FINAL_AMOUNT DESC;
```

8. Finding all the passengers travelling in the month of October.

```
SELECT
FNAME,
   LNAME
FROM
USER
WHERE
EMAIL IN
   (
SELECT
USER_EMAIL
FROM
BUS_TRIP
WHERE
DEPART_TIME >= '20201001' AND DEPART_TIME < '20201101'
)
ORDER BY FNAME DESC, LNAME DESC;
```

9. To find the list of all the passengers who have spent more than 1000 for their TRIP along with the source and destination stations.

```
SELECT
PNR,
   DEPART_STATION AS DEPARTING_STATION,
   ARRIVAL_STATION AS DESTINATION,
   ARRIVAL_TIME AS ARRIVAL
FROM
BUS_TRIP
WHERE
PNR IN
   (
SELECT
PNR
FROM
FARE
WHERE
TOTAL >= 1000
);
```

Conclusion

An efficient database management is important for every industry, especially the tourism industry where n numbers of transactions are happening every minute. This **Intra Indian Bus Transit System (IIBTS)** database that has been created, tries to automate the whole process. There are a few limitations and constraints that are involved where future enhancement can be done.

LIMITATION

- The more complex operations like refund on a cancellation are not done in this project.
- In an ideal world the details of the drivers also should be provided to the user which has not been done in this project.
- Many more entities such as insurance, travel route-stop details can be added.
  These are some of the limitations

FUTURE ENHANCEMENT

- Application of GUI and front end software.
- Airport like web check in system could be emulated.
- Live bus tracking details could be stored in the database and then shared with the Passengers.
- ETA could be stored in the database and then shared with the Passengers